PCT

WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 7 : G06F 11/00

A1

(11) International Publication Number: WO 00/68790

(43) International Publication Date: 16 November 2000 (16.11.00)

(21) International Application Number: PCT/IL99/00236

(22) International Filing Date: 6 May 1999 (06.05.99)

(71) Applicant *(for all designated States except US)*: REALITY BYTES R & D LTD. [IL/IL]; Yehudit Boulevard 35, 67016 Tel Aviv (IL).

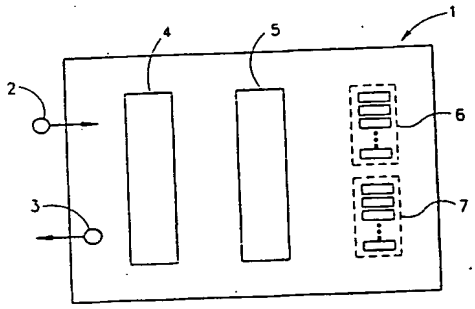
(72) Inventor; and
(75) Inventor/Applicant *(for US only)*: COHEN, Nadav [IL/IL]; 30046 Moshav Beit Shearim (IL).

(74) Agent: REINHOLD COHN AND PARTNERS; P.O. Box 4060, 61040 Tel Aviv (IL).

(81) Designated States: AE, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, CA, CH, CN, CU, CZ, DE, DK, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MD, MG, MK, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, UA, UG, US, UZ, VN, YU, ZA, ZW, ARIPO patent (GH, GM, KE, LS, MW, SD, SL, SZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

Published
With international search report.

(54) Title: AN OPERATING SYSTEM INTERFACE METHOD



(57) Abstract

In a systems–architecture having a processor for executing application programs or kernel modules, an operating system interface method including the steps of: providing a hook manager for managing hooking and unhooking of system calls by application programs or kernel modules; and passing control to at least one hook before a system call or after a system response. The manager provides a convenient way to externally expand services provided by an operating system, or to externally resolve incompatibilities between an application program and an operating system.

FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AL	Albania	ES	Spain	LS	Lesotho	SI	Slovenia
AM	Armenia	FI	Finland	LT	Lithuania	SK	Slovakia
AT	Austria	FR	France	LU	Luxembourg	SN	Senegal
AU	Australia	GA	Gabon	LV	Latvia	SZ	Swaziland
AZ	Azerbaijan	GB	United Kingdom	MC	Monaco	TD	Chad
BA	Bosnia and Herzegovina	GE	Georgia	MD	Republic of Moldova	TG	Togo
BB	Barbados	GH	Ghana	MG	Madagascar	TJ	Tajikistan
BE	Belgium	GN	Guinea	MK	The former Yugoslav Republic of Macedonia	TM	Turkmenistan
BF	Burkina Faso	GR	Greece			TR	Turkey
BG	Bulgaria	HU	Hungary	ML	Mali	TT	Trinidad and Tobago
BJ	Benin	IE	Ireland	MN	Mongolia	UA	Ukraine
BR	Brazil	IL	Israel	MR	Mauritania	UG	Uganda
BY	Belarus	IS	Iceland	MW	Malawi	US	United States of America
CA	Canada	IT	Italy	MX	Mexico	UZ	Uzbekistan
CF	Central African Republic	JP	Japan	NE	Niger	VN	Viet Nam
CG	Congo	KE	Kenya	NL	Netherlands	YU	Yugoslavia
CH	Switzerland	KG	Kyrgyzstan	NO	Norway	ZW	Zimbabwe
CI	Côte d'Ivoire	KP	Democratic People's Republic of Korea	NZ	New Zealand		
CM	Cameroon			PL	Poland		
CN	China	KR	Republic of Korea	PT	Portugal		
CU	Cuba	KZ	Kazakstan	RO	Romania		
CZ	Czech Republic	LC	Saint Lucia	RU	Russian Federation		
DE	Germany	LI	Liechtenstein	SD	Sudan		
DK	Denmark	LK	Sri Lanka	SE	Sweden		
EE	Estonia	LR	Liberia	SG	Singapore		

An Operating System Interface Method

FIELD OF THE INVENTION

This invention relates to a method for interfacing computer applications to a computer's operating system.

BACKGROUND OF THE INVENTION

The internal operational organization of today's computer is the culmination of numerous simultaneous engineering considerations. These considerations have brought an ensemble of organizational tools to the task of developing and maintaining software constructed systems. Simultaneously, there is evolving in the computer environment mutual design considerations for computational efficiency, systems security, functional modularity for increasing maintenance and compatibility, etc.

One area, of the presently accepted software architecture design, which has been held in a relatively static state, is the interfacing between a central operating system and peripheral applications programs. Given the present complexity of operating systems and of applications programs, adding of functions to either an operating system or to an application program is a complex undertaking, which often requires global systems-scale testing procedures. There is a need in the art for a method of adding functions in a way that does not require such a cumbersome level of testing. Furthermore, there is a need in the art for a method of interfacing with an operating system in a way that does not contribute to the complexity of such testing.

SUMMARY OF THE INVENTION

The present invention relates to an operating system interface method, for use in a systems-architecture having a processor for executing application programs or kernel modules. This method includes the steps of: **providing** a hook manager for managing hooking and unhooking of system calls by application programs or kernel modules; and **passing** control to at least one hook before a system call or after a system response.

Operating systems are designed to run on a specific computer architecture. In the context of the preferred embodiment of the present invention, an operating system is a substantially centralized software entity running on an externally monolithic aggregate of a computer and its associated peripheral devices (e.g. printer, display, etc.). In the context of other embodiments of the present invention, an operating system is a distributed ensemble of like or linked software entities strategically proliferated into a distributed processing architecture (e.g. a shuffle exchange, a hyper cube, an array processor, a client server network system, etc.).

In each respective architecture, the operating system receives system calls from application programs or from kernel modules. Likewise, the operating system directs system responses to applications programs or to kernel modules. Accordingly, a hook manager is provided, by the present invention, to evaluate an aspect of each system call or system response. The evaluation takes each intercepted call or response, and either passes it on to its expected destination or diverts it to an alternative destination.

By this evaluation and diversion, a general use interfacing with an operating system is provided in a way that does not contribute to the complexity of either operating system testing or applications program testing.

From the vantage of the operating system, the manager is just another applications program. From the vantage of an applications program, the manager is the operating system. Most importantly, from the vantage of a software developer, the manager provides a convenient way to externally expand the services provided

by an operating system, and to externally resolve incompatibilities between an application program and an operating system.

According to one common scenario, features of an applications program are specifically built to conform to a special feature of an operating system. Thereafter, there is a desire to use this applications program with another operating system that does not include this special feature. According to the present invention, the manager intercepts all system calls. When the system call is for the special feature, the manager diverts the call to an external "special feature emulator".

According to another common scenario, there is a need to analysis the efficiency of an operating system is a specific environment. According to the present invention, the manager diverts a specific class of system calls to a call-log application where these systems calls are recorded, and thereafter routed via the manager to their original destination, the operating system. An applications program is thereby provided data, e.g. the call-log record, for the needed analysis.

Returning to the issue of defining a "manager" with respect to a specific computer architecture environment, the provided manager may be a single centralized entity operating as an intermediary to the operating system of an externally monolithic aggregate of a computer and its associated peripheral devices. For an operating system that is a distributed ensemble of like or linked software entities strategically proliferated into a distributed processing architecture, the manager may a like numbered ensemble of co-managers assigned one co-manager per operating system entity.

Not unlike human management models, the "software" manager of the present invention may create and assign task specific, process specific, or call specific "supervisor" software entities to any whole or partial application program, or even to any group of application programs.

Those versed in the art should appreciate that ostensibly the present invention does not provide any new functionality to the overall computer-hardware-software conglomerate. Rather, the present invention allows for many of the known or desired functions to be incorporated into the conglomerate, in an

independent modular manner. This facilitates the upward compatibility of both operating systems and applications programs. This facility is especially cost efficient when operating systems or applications programs embody special features, as is commonly the case. Since it is never known in advance which of these features will survive as industry standards and which will lapse into oblivion, the present invention provides an efficient facile solution for those who have invested man-years into utilizing options destined for oblivion.

Furthermore, the present invention provides a rapid development track for the marketing of new system-like utility features, which have not heretofore been incorporated into operating systems. This rapid development track may prove to be useful to the developers of standard accepted operating systems, as a means for introducing new features and for testing their acceptance in the market. This rapid development track may also prove to be useful to small specialty-type developers, who provide peculiar emulation or conversion software routines to clients switching from one operating system to a different operating system.

BRIEF DESCRIPTION OF THE DRAWINGS

In order to understand the invention and to see how it may be carried out in practice, a preferred embodiment will now be described, by way of non-limiting example only, with reference to the accompanying drawings, in which:

Fig. 1 is a block diagram of a systems-architecture; and

Fig. 2 is a flowchart of a complete system call and system response cycle.

DETAILED DESCRIPTION OF A PREFERRED EMBODIMENT

Fig. 1 shows a block diagram of a systems-architecture wherein a processor **1** has an input modality interface **2** and an output modality interface **3**. The processor includes an operating system **4**, a hook manager **5**, a set of application programs **6**, and a set of kernel modules **7**.

The present invention relates to an operating system interface method, for use in a systems-architecture having a processor for executing application programs or kernel modules. This method includes the steps of:

(a) **providing** a hook manager for managing hooking and unhooking of system calls by application programs or kernel modules; and

(b) **passing** control to at least one hook before a system call or after a system response.

According to an embodiment of the present invention, passing control includes copying additional information to at least one hook. Examples of "additional information" include systems state information, process memory information, kernel tables, response status, etc.

According to an embodiment of the present invention, hooking includes calling an application program interface (API) to receive system calls or to no longer receive system calls.

According to an embodiment of the present invention, the hook manager includes an application program interface (API) allowing receiving of system responses or discontinuing receiving of system responses.

According to an embodiment of the present invention, managing includes using the hook manager for:

(a) building a chain of hooks in response to requests by an application program or kernel module to receive system calls; and

(b) removing a hook from said chain of hooks, in response to requests by an application program or kernel module to no longer receive system calls.

In the context of the present invention, a "chain" is an organization or structure such as a list, a stack, a tree, a net-like graph, etc.

According to one variation of the present invention, building a chain of hooks is in sequential order based on when the requests are received. According to another variation of the present invention, removing a hook is in a non-last-in-first-out (non-LIFO) fashion.

According to an embodiment of the present invention, the hook manager is an application program or a kernel module or a part thereof.

According to an embodiment of the present invention, passing control includes an interim step selected from the list:

(a) a hook under management of a hook manager gaining access to the system call or system response;

(b) logging the system call or system response;

(c) filtering the system call or system response;

(d) encrypting or decrypting the system call or system response;

(e) compressing or decompressing the system call or system response;

(f) modifying, or altering, or replacing the system call or system response; or

(g) delaying the system call or system response.

Fig. 2 shows a flowchart of a complete system call and system response cycle. Starting at a "BEGIN" **20**, a query step "ARE THERE ANY PRE-CALL HOOKS?" **21** is imposed. If there are pre-call hooks, then "GET A HOOK FROM THE HOOK CHAIN" **22**. "IF THIS IS THE END OF THE CHAIN" **23**, then make the original call at **24** to the operating system **4**; otherwise perform optional tests or processing at **25** and return to **22**. "Optional tests or processing" may include checking for permissions, setting a processing delay or priority, etc.

Symmetrically, but starting from an operating system response to entry point **30**, initiates a post processing a query step "ARE THERE ANY POST-CALL HOOKS?" **31**. If there are post-call hooks, then "GET A HOOK FROM THE HOOK CHAIN" **32**. "IF THIS IS THE END OF THE CHAIN" **33**, then return to **40**; otherwise perform optional tests or processing at **35** and return to **32**. Again, "optional tests or processing" may include checking for permissions, setting a processing delay or priority, etc.

The present invention relates to an operating system interface method, for use in a systems-architecture having a processor for executing application programs or kernel modules, the method comprising the steps of:

(a) intercepting a system call **21** made by an application program or by a kernel module;

(b) in a hook chain, determining **22** which of the hooks is permitted to receive control before or after the system call;

(c) if there is at least one determined hook, then selecting at least one of the determined hooks; and

(d) passing control to the at least one hook of the selected hooks.

The present invention also relates to an operating system interface method, for use in a systems-architecture having a processor for executing application programs or kernel modules, the method comprising the steps of:

(a) intercepting a system response **31** made to an application program or to a kernel module;

(b) in a hook chain, determining **32** which of the hooks is permitted to receive control before or after the system response;

(c) if there is at least one determined hook, then selecting at least one of the determined hooks; and

(d) passing control to the at least one hook of the selected hooks.

It should be appreciated that a system call and a system response are substantially equivalent with respect to functionality and complexity, in the context of the present invention.

In the context of the present invention, the expression "determined hook" relates to the notion "determining" using an algorithm, as is commonly used in operating systems security, scheduling, etc. For example, determining to assign, verify, or certify a most recent state of, status of, modification of, or systems characterization of a system related user, file, or resource.

According to an embodiment of the present invention, intercepting includes providing a hook manager. Furthermore, according to one variation of the present invention, the hook manager includes an application program interface (API). According to another variation of the present invention, the hook manager is an application program or a kernel module or a part thereof.

According to the preferred embodiment of the present invention, passing control includes an interim step selected from the list:

(a) a hook under management of a hook manager gaining access to the system call or a system response;

(b) logging the system call or a system response;

(c) filtering the system call or a system response;

(d) encrypting or decrypting the system call or a system response;

(e) compressing or decompressing the system call or a system response;

(f) modifying, or altering, or replacing the system call or a system response; or

(g) delaying the system call or a system response.

Furthermore, the present invention relates to an operating system **4** interface-apparatus comprising systems-architecture having a processor **1** for executing application programs **6** or kernel modules **7**, wherein said systems-architecture includes:

- a hook manager **5** for managing hooking and unhooking of system calls by application programs or kernel modules: and
- a controller (in **1** or **4**) for passing to at least one hook before a system call or after a system response.

In the method claims which follow, alphabetic characters used to designate claim steps are provided for convenience only and do not imply any particular order of performing the steps.

CLAIMS:

1. An operating system interface method, for use in a systems-architecture having a processor for executing application programs or kernel modules, the method comprising the steps of:

(a) **providing** a hook manager for managing hooking and unhooking of system calls by application programs or kernel modules; and

(b) **passing** control to at least one hook before a system call or after a system response.

2. The method according to claim 1 wherein passing control includes copying additional information to at least one hook.

3. The method according to claim 1 wherein hooking includes calling an application program interface (API) to receive system calls.

4. The method according to claim 1 wherein unhooking includes calling an application program interface (API) to no longer receive system calls.

5. The method according to claim 1 wherein the hook manager includes an application program interface (API) allowing receiving of system responses.

6. The method according to claim 1 wherein the hook manager includes an application program interface (API) discontinuing receiving of system responses.

7. The method according to claim 1 wherein managing includes using the hook manager for:

(a) building a chain of hooks in response to requests by an application program or kernel module to receive system calls; and

(b) removing a hook from said chain of hooks, in response to requests by an application program or kernel module to no longer receive system calls.

8. The method according to claim 7 wherein building a chain of hooks is in sequential order based on when the requests are received.

9. The method according to claim 7 wherein removing is in a non-last-in-first-out (non-LIFO) fashion.

10. The method according to claim 1 wherein the hook manager is an application program or a kernel module or a part thereof.

11. The method according to claim 1 wherein passing control includes an interim step selected from the list:

(a) a hook under management of a hook manager gaining access to the system call or system response;

(b) logging the system call or system response;

(c) filtering the system call or system response;

(d) encrypting or decrypting the system call or system response;

(e) compressing or decompressing the system call or system response;

(f) modifying, or altering, or replacing the system call or system response; or

(g) delaying the system call or system response.

12. An operating system interface method, for use in a systems-architecture having a processor for executing application programs or kernel modules, the method comprising the steps of:

(a) intercepting a system call made by an application program or by a kernel module;

(b) in a hook chain, determining which of the hooks is permitted to receive control before or after the system call;

(c) if there is at least one determined hook, then selecting at least one of the determined hooks; and

(d) passing control to the at least one hook of the selected hooks.

13. The method according to claim 12 wherein intercepting includes providing a hook manager.

14. The method according to claim 13 wherein the hook manager includes an application program interface (API).

15. The method according to claim 13 wherein the hook manager is an application program or a kernel module or a part thereof.

16. The method according to claim 12 wherein passing control includes an interim step selected from the list:

(a) a hook under management of a hook manager gaining access to the system call or a system response;

(b) logging the system call or a system response;

(c) filtering the system call or a system response;

(d) encrypting or decrypting the system call or a system response;

(e) compressing or decompressing the system call or a system response;

(f) modifying, or altering, or replacing the system call or a system response; or

(g) delaying the system call or a system response.

17. An operating system interface method, for use in a systems-architecture having a processor for executing application programs or kernel modules, the method comprising the steps of:

(a) intercepting a system response made to an application program or to a kernel module;

(b) in a hook chain, determining which of the hooks is permitted to receive control before or after the system response;

(c) if there is at least one determined hook, then selecting at least one of the determined hooks; and

(d) passing control to the at least one hook of the selected hooks.

18. The method according to claim 17 wherein intercepting includes providing a hook manager.

19. The method according to claim 18 wherein the hook manager includes an application program interface (API).

20. The method according to claim 18 wherein the hook manager is an application program or a kernel module or a part thereof.

21. The method according to claim 17 wherein passing control includes an interim step selected from the list:

(a) a hook under management of a hook manager gaining access to a system call or the system response;

(b) logging a system call or the system response;

(c) filtering a system call or the system response;

(d) encrypting or decrypting a system call or the system response;

(e) compressing or decompressing a system call or the system response;

(f) modifying, or altering, or replacing a system call or the system response or

(g) delaying a system call or the system response.

22. An operating system interface-apparatus comprising systems-architecture having a processor for executing application programs or kernel modules, wherein said systems-architecture includes:

(a) a hook manager for managing hooking and unhooking of system calls by application programs or kernel modules; and

(b) a controller for passing to at least one hook before a system call or after a system response.

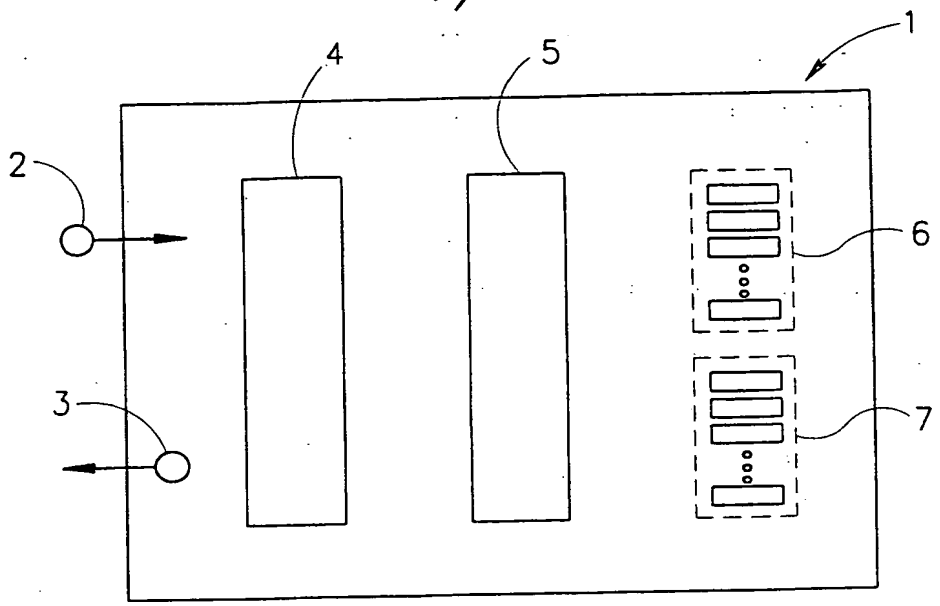

1
4
5
2
6
3
7

FIG.1

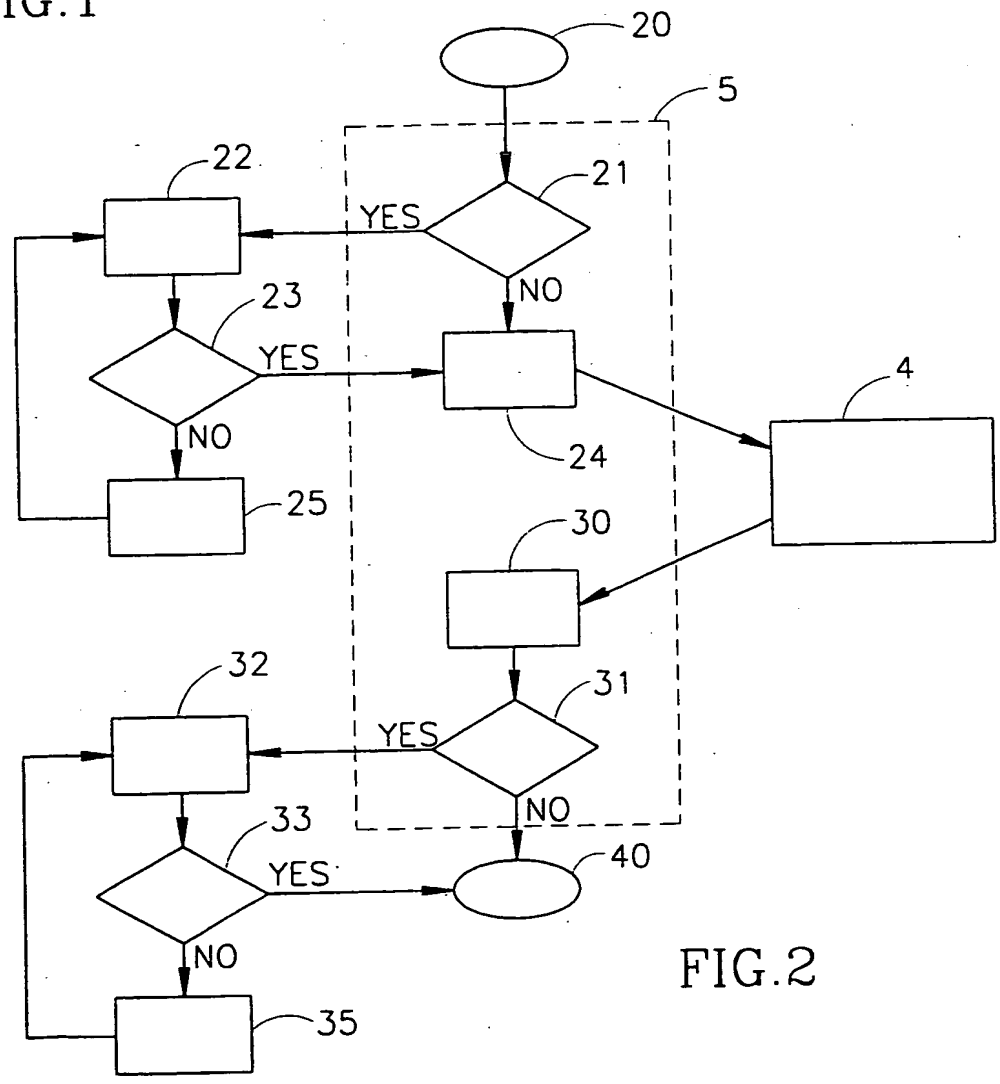

20
5
22
21
YES
NO
23
YES
4
NO
24
25
30
32
31
YES
NO
33
YES
40
NO
35

FIG.2

INTERNATIONAL SEARCH REPORT

International Application No
PCT/IL 99/00236

A. CLASSIFICATION OF SUBJECT MATTER
IPC 7 G06F11/00

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
IPC 7 G06F

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category °	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	RUSSINOVICH M ET AL: "EXAMINING VXD SERVICE HOOKING" DR. DOBB'S JOURNAL, vol. 21, no. 5, May 1996 (1996-05), pages 32,34,36-37, XP002035779 US,SAN MATEO, CA the whole document ---	1-22
X	MRKOR, KAI-UWE: "NACHGEHAKT WINDOWS-NACHRICHTEN FILTERN MIT HOOK-FUNTIONEN" C'T MAGAZIN FÜR COMPUTER TECHNIK, no. 5, page 272-274,276,278,280-281 XP000800884 DE,VERLAG HEINZ HEISE GMBH., HANNOVER ISSN: 0724-8679 the whole document ---	1-22

-/--

[X] Further documents are listed in the continuation of box C.

[X] Patent family members are listed in annex.

° Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
22 December 1999	13/01/2000

Name and mailing address of the ISA	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Tx. 31 651 epo nl, Fax: (+31-70) 340-3016	Wiltink, J

Form PCT/ISA/210 (second sheet) (July 1992)

INTERNATIONAL SEARCH REPORT

Inter . .ional Application No
PCT/IL 99/00236

C.(Continuation) DOCUMENTS CONSIDERED TO BE RELEVANT

Category *	Citation of document, with indication,where appropnate, of the relevant passages	Relevant to claim No.
A	JONES M B: "Interposition agents: transparently interposing user code at the system interface" 14TH ACM SYMPOSIUM ON OPERATING SYSTEMS PRINCIPLES, ASHVILLE, NC, USA; OPERATING SYSTEMS REVIEW, 'Online! vol. 27, no. 5, 5 - 8 December 1993, pages 80-93, XP002126575 USA ISSN: 0163-5980 ACM Digital Library Retrieved from the Internet: <URL:http://www.acm.org/pubs/articles/proceedings/ops/168619/p80-jones/p80-jones.pdf > 'retrieved on 1999-12-22! abstract page 80, right-hand column, line 14 -page 83, left-hand column, last line	1-22
A	US 5 454 086 A (ALPERT ALAN I ET AL) 26 September 1995 (1995-09-26) abstract column 2, line 45 -column 8, line 2; figures 1-4	1-22

INTERNATIONAL SEARCH REPORT

Information on patent family members

Inter. ..ional Application No
PCT/IL 99/00236

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 5454086 A	26-09-1995	JP 2539161 B	02-10-1996
		JP 7093183 A	07-04-1995